Motor Control Designs with an Integrated FPGA Design Flow

Thomas Chau, System Solution Engineering, Altera, Now Part of Intel
Ben Jeppesen, System Solution Engineering, Altera, Now Part of Intel
Kevin Smith, System Solution Engineering, Altera, Now Part of Intel
Andrew Crosland, System Solution Engineering, Altera, Now Part of Intel
Stefano J. Zammattio, Product Marketing, Altera, Now Part of Intel

WP-01162-2.0 White Paper

This white paper describes a recommended design flow that leverages Altera® FPGAs' adaptability, variable-precision digital signal processing (DSP), and integrated system-level design tools for motor control designs. Designers of motor-driven equipment can take advantage of the performance, integration, and efficiency benefits of this design flow.

Introduction

Altera's FPGA architectures provide an effective platform for motor drive systems because of the following advantages:

- Design integration—Integrate an embedded processor, encoder interfacing, DSP motor control algorithms, and industrial networking in a single device.
- Performance and deterministic latency—Achieve higher performance and efficiency on different types of motors, and support motor control algorithms that require deterministic operations.
- Streamlined design flow—Use model-based design tools such as Simulink, combined with Altera's DSP Builder and Qsys to optimize the full motor system in a low-cost FPGA. Reuse intellectual property (IP) and take advantage of variable-precision DSP blocks. Use fixed- or floating-point precision for any part of the control path.

Although it is common to use off-the-shelf microcontroller units or DSP devices to implement processing and control loops that monitor load and adjust position, velocity and other drive aspects, these devices have a number of limitations, such as fixed memory, limited I/O, and switching transistor modulation limited to predefined pulse width modulation (PWM). Microcontroller units (MCUs) are particularly limited by their lack of scalability and performance. These deficiencies are most evident in systems of increasingly complex algorithms with high millions-of-instructions-per-second (MIPS) processing requirements. While high-end DSP devices typically have the power to handle motor control computations, they are not ideal in a system that simultaneously incorporates time-precise operations with task-oriented operations, such as memory interfacing, signal interfacing and filtering, and supporting an industrial Ethernet protocol standard.

Design Integration

Next-generation drives that require more performance and improved motor efficiencies require a platform that provides the flexibility to integrate and optimize the system. Using Altera FPGAs, designers can embed multiple processors or use the flexible DSP capabilities, and then leverage additional logic, custom instructions, or one of the many supported industrial networking protocols. Altera FPGAs enable designers to implement multiple embedded processors to control each subsystem independently. The parallel nature of Altera FPGAs supports integration of most motor control system building blocks. For example, Altera's Nios® II embedded processor (32 bit RISC soft processor) can control all of the various interfaces and sensors and encoders. Designers can then use variable-precision, floating-point DSP blocks to perform field-oriented control (FOC) or other math-intensive algorithms.

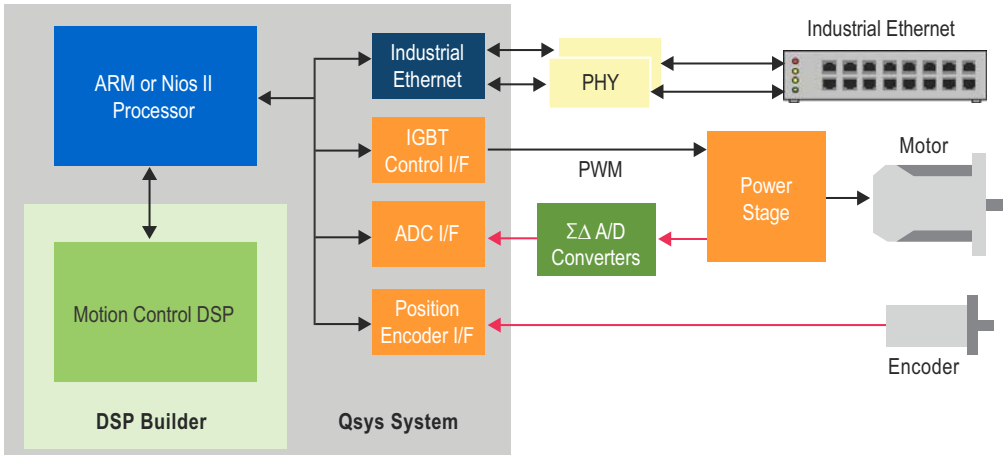
Figure 1 illustrates the variety of elements that can be integrated in the FPGA to create a single drive-on-a-chip system. Integrated IP functions can run in parallel, ensuring that there are no bottlenecks in either sequential or time-delayed operations.

This design flow supports integration of useful IPs, including the following:

- Position feedback—Encoders with high-precision position feedback, such as EnDAT and Biss, allow 10X faster speed and position data decoding.
- Control of switching transistors—IGBTs or MOSFETs switch the DC link voltage to drive AC motors. Space vector modulation (SVM) controls the gate input of the switching transistors to generate the sinusoidal voltage wave necessary to drive the motor.
- ADC interface—An internal or external analogue-to-digital converter (ADC) measures current feedback from the motor. Sigma-delta (ΣΔ) ADCs are easier to isolate from high drive voltages, have lower noise, and support sampling of their outputs by the FPGA to give fast and accurate readings.
- Networking interface – Industrial Ethernet is becoming a more common feature in industrial drives. Real-time protocols in the FPGA accommodate the industrial Ethernet protocol standards required for the application, such as Ethernet/IP, PROFINET IO/IRT and EtherCAT.

The proliferation of these DSP-based motor control functions, communications, and interface standards make FPGAs an ideal platform for industrial motor drives.

Figure 1. Optimized Motor Control FPGA Design Flow

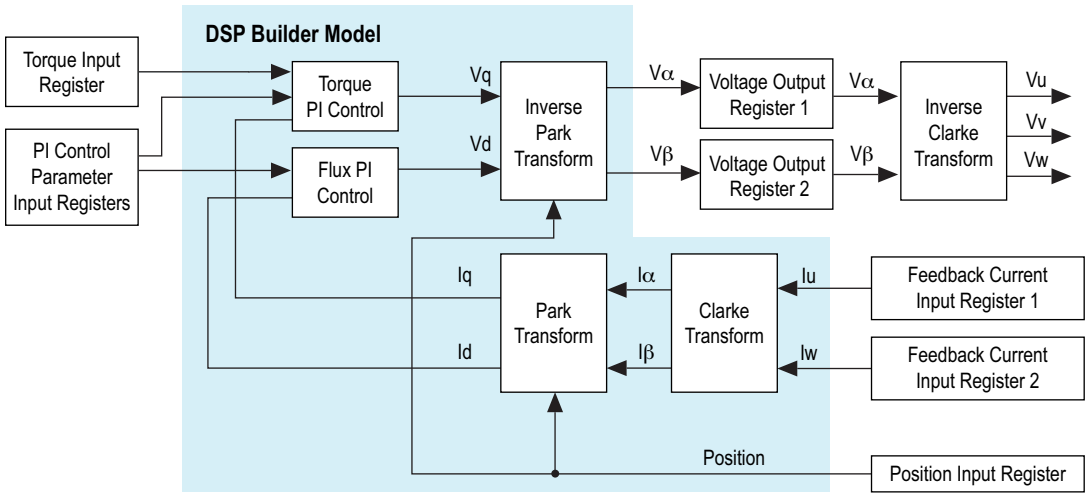


Performance and Deterministic Latency

FPGAs can easily scale performance based on the application requirements. Altera FPGAs, with the industry's first variable-precision DSP block, provide the flexibility to choose the precision level that exactly matches the requirements, and also supports standard single- or double-precision floating-point types. These factors make the DSP block an ideal choice for implementing complex math algorithms. The integrated DSP block—a feature in many of Altera's FPGA architectures—allows configuration of each block at compile time. In addition, Altera's DSP Builder tool reports hardware latency in the design stage before running in hardware, thus facilitate designs which deterministic latency is crucial.

Field-oriented control (FOC, see Figure 2), also known as vector control, is commonly used to control permanent magnet synchronous motors (PMSM). FOC is useful in industrial servo motors that require precise torque control. FOC techniques help to reduce motor size, cost, and power consumption. In addition, FOC reduces torque ripple and electromagnetic interference. However, running the algorithm at high frequency requires significant computing power.

Figure 2. FOC Model for Motor Control



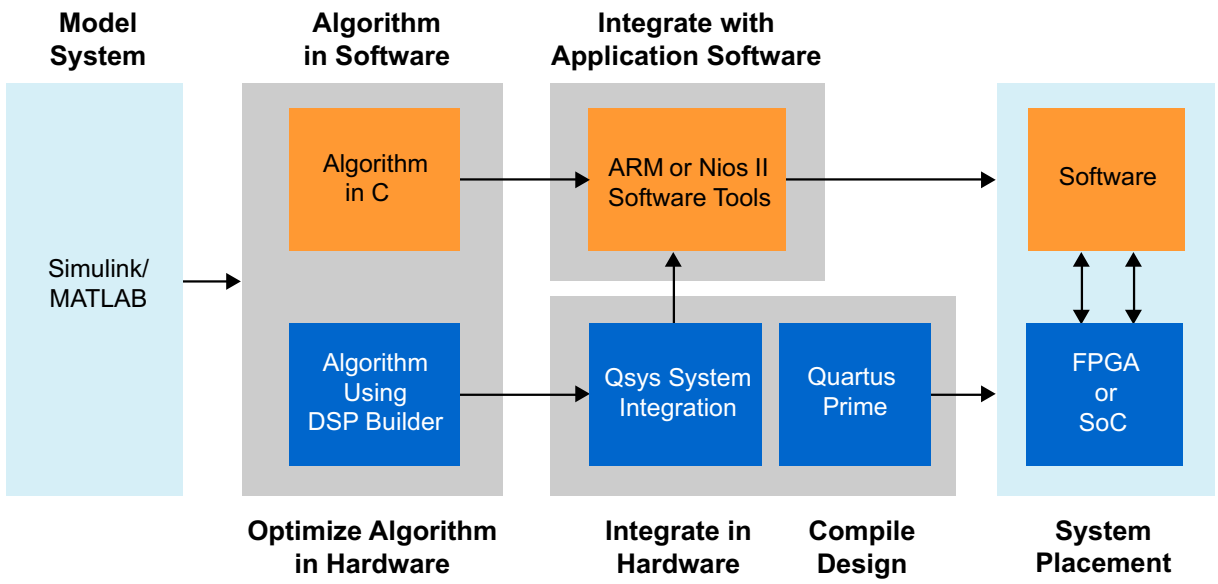
FOC aligns the vector of motor phase currents to produce the required torque with the minimum current magnitude and therefore reduces power consumption. The algorithm involves the following steps:

1. Convert the 3-phase feedback current inputs and the rotor position from the encoder into quadrature and direct current components using Clarke and Park transforms.
2. Use these current components as the inputs to two proportional and integral (PI) controllers running in parallel to limit the direct current to zero and the quadrature current to the desired torque.
3. Convert the direct and quadrature current outputs from the PI controllers back to 3-phase currents with inverse Clarke and Park transforms.

Streamlined Design Flow

Developing motor control designs requires versatile tools (and a practical tool flow) to help model and simulate the system, implement complex algorithms with low latency, and have the ability to integrate the system together and fine tune the performance to the exact needs of the motor drive. Model-based design (shown in Figure 3) allows designers to reduce development time, and provide a more flexible, powerful model that is scalable for different types of drive systems.

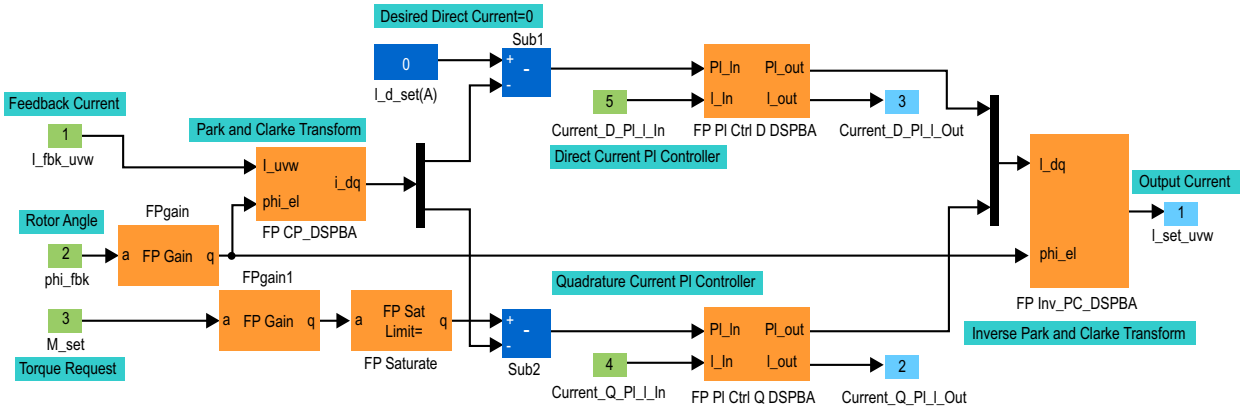
Figure 3. Optimized Motor Control FPGA Design Flow



Altera provides embedded designers with powerful and easy-to-use development tools, such as the Quartus® Prime design software and MegaCore® IP library. Altera also provides the Qsys system integration tool Qsys, and DSP Builder for DSP optimization. In addition, Altera offers the Eclipse-based Embedded Design Suite (EDS) for programming the Nios II or ARM®-based processor.

Altera's DSP Builder is a Simulink blockset which allows simulation of FPGA DSP functions within Simulink, automatic generation of hardware description language (HDL) and automated testing with ModelSim®-Altera software. For example, DSP Builder allows designers to model the FOC algorithm directly in MATLAB/Simulink by constructing a block diagram that represents their system, as shown in Figure 4. Depending on the FPGA and CPU resource utilization, designers can partition these elements between a hardware and software implementation.

Figure 4. FOC Model Implemented in Simulink



Simulink allows designers to run bit-accurate mathematical simulations of the behaviour of the algorithm against a model or system. When designers have finished developing the algorithm, designers can optimize an FPGA-based DSP system design with DSP Builder as listed below:

- Perform optimizations such as pipelining and resource sharing to produce an efficient register transfer level (RTL) representation optimized for the chosen Altera FPGA.
- Model the algorithm with fixed-point, single- or double-precision floating-point, or custom-precision floating point types. Designers can initially model the algorithm in Simulink using a higher precision than necessary, and then scale the precision within the tool for final implementation. Designers are able to directly observe run-time latency, data throughput, and algorithm usage results in Simulink before running in hardware, so they can perform design space exploration with Simulink to choose the most suitable implementation.
- Apply automatic 'folding', which is closely related to time-division multiplexing (TDM). DSP Builder's folding functionality can automatically take advantage of the difference between the system clock rate and the data rate to optimize the use of FPGA hardware. When the system clock rate is faster than the data or sample rate, a single hardware component (such as a multiplier) can potentially process multiple data points, thus resulting in FPGA resource savings. Similarly, in a system with multiple parallel data sources or data channels, instead of duplicating hardware for each channel or data source, designers can use one datapath to process multiple data channels. Designers can implement the core algorithm in the most intuitive way, as if there were no folding applied. By tuning the folding factor in DSP Builder, designers can trade off the throughput, resource usage, and latency of the generated logic without requiring redesign.

For more information about DSP Builder, including the standard and advanced blocksets, refer to the *DSP Handbook* web page.

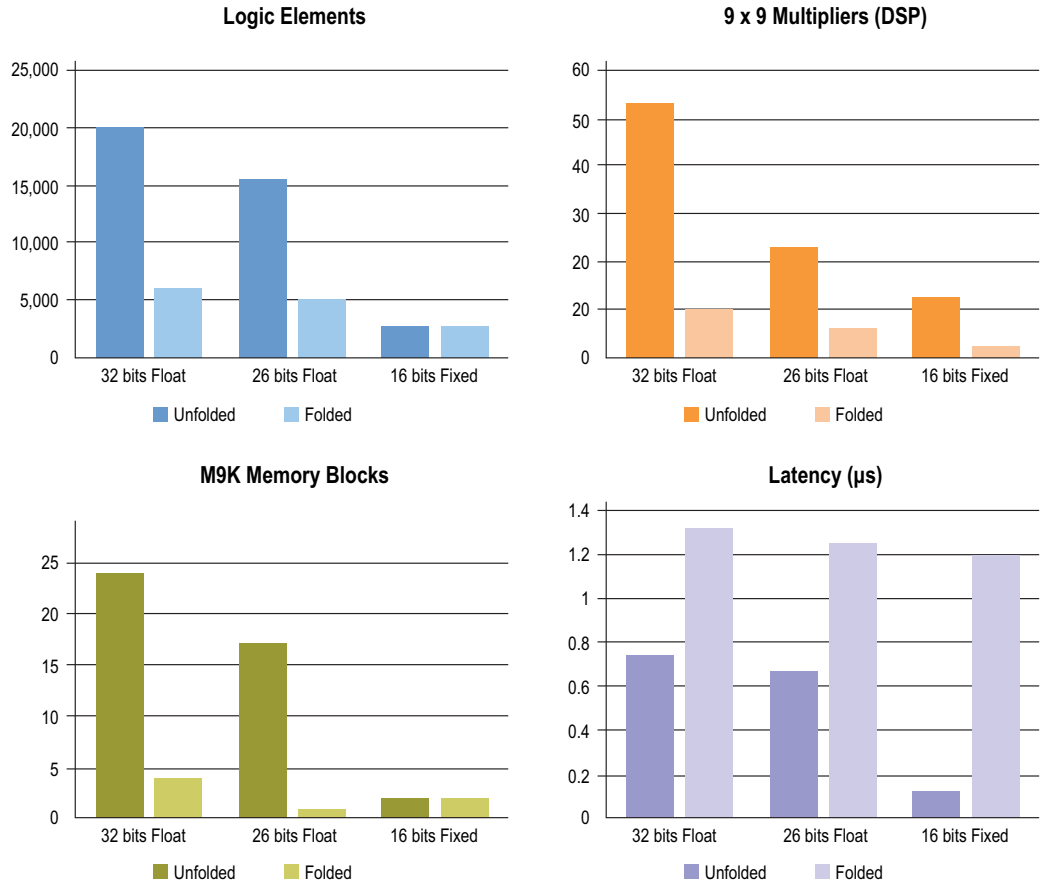
Benchmark Results

To highlight these features and functions, Altera evaluated the performance of different implementations of FOC in the DSP Builder software. In a typical FOC controller the inputs are sampled at 10-100 ksps, a rate easily handled by the 100-MHz FPGA clock rate. At 100 ksps, a new sample must be processed every 10 µs. Keeping the processing latency constant and to a minimum helps the performance of the control algorithm.

The DSP Builder software allows both fixed- and floating-point implementations. The precision is controlled using MATLAB workspace variables so designers can perform simple experiments. The following section shows the results achieved using single-precision (32 bit) and custom-precision (26 bit) floating-point types and 16 -bit fixed-point, all implemented in a MAX® 10 device with 100 MHz clock.

By default, the DSP Builder software creates a fully pipelined RTL representation that can accept new input values every clock cycle. The result obtained for this "unfolded" configuration was then compared to a "fully folded" configuration. Figure 5 summarizes the results.

Figure 5. Systems Resources and Latency Comparison for MAX 10 Devices



The benchmarking experiments illustrate the following points:

- Compared to standard floating-point precision, variable floating-point precision provides flexibility in terms of resource usage and latency as datapaths are narrower and simpler with reduced precision.
- Fixed-point arithmetic uses significantly fewer resources than floating-point. Typically a design that does not require the high dynamic range afforded by floating-point is implemented in fixed point. However, floating point avoids arithmetic overflow during algorithm development and tuning.
- Fixed-point models can achieve a processing latency down to 0.1µs, ideal for designs that require very high update frequencies.
- Folded models use significantly fewer resources than unfolded ones. Although folding increases latency to around 1µs, this is still acceptable for control loops updating at 100-ksps or more.

Conclusion

Today's modern MCUs and DSP devices may not be suitable for next-generation motor control systems. Designers require the flexibility to fine-tune motor control algorithms and transistor switching patterns to reduce control latency and power consumption. Off-the-shelf DSP solutions have limited fixed-point or floating-point processing capabilities, fixed PWM IP and may not include other components required to drive the system.

Altera FPGAs allow integration of components, such as a processor that can manage the overall operation, flexible interfacing to easily connect to other systems, acceleration of control algorithms and a model-based design flow with automatic code generation. Altera FPGAs' inherent parallel processing capabilities and variable-precision DSP blocks enable you to achieve synchronized control of multiple motors and balance resource usage against performance requirements. These advantages together enable you to design the most sophisticated drive systems.

Further Information

- Altera in Industrial
  www.altera.com/end-markets/industrial/ind-index.html
- Drive-On-Chip Reference Design
  www.altera.com/en_US/pdfs/literature/an/an669.pdf
- DSP Builder Handbook
  www.altera.com/literature/technology/dsp/lit-dsp.jsp
- White Paper: *Lowering the Total Cost of Ownership in Industrial Applications*
  www.altera.com/literature/wp/wp-01122-tco-industrial.pdf
- White Paper: *A Flexible Solution for Industrial Ethernet*
  www.altera.com/literature/wp/wp-01037.pdf
- White Paper: *Developing Functionally Safe Systems with TÜV-Qualified FPGAs*
  www.altera.com/literature/wp/wp-01123-functional-safety.pdf

Document Revision History

Table 1 shows the revision history for this document.

Table 1. Document Revision History

Date	Version	Changes
May 2016	2.0	Updated content and graphics.
May 2012	1.2	Minor text and formatting edits.
July 2011	1.1	Minor text edits.
April 2011	1.0	Initial release.